Amendments to the Specification:

Please **replace** the title of the application with the following amended title:

AUTOMATED RECOVERY FROM DATA CORRUPTION OF DATA VOLUMES IN PARITY RAID STORAGE SYSTEMS

Please **replace** paragraph **[0011]** with the following amended paragraph:

[0011] Another form of data corruption can occur if disk drive's firmware doesn't write the data to the disk platter, but reports successful completion of the write. In that case, the data stored in the disk block may be internally consistent, but "stale" and **[there for]** **therefore** considered corrupted.

Please **replace** paragraph **[0024]** with the following amended paragraph:

[0024] The first error correction data may take any one of many different forms. Whatever form the first error correction takes, data of any stripe unit $B_{y,x}$ can be generated as a function of data in stripe units $B_{y,1}$ - $B_{y,4}$, other than stripe unit $B_{y,x}$, and the first error correction data of stripe unit SP_y, when, for example, data of stripe unit $B_{y,x}$ is inaccessible due **to a** hardware or software failure. In one embodiment, each stripe unit SP_y stores parity data calculated as a function of data in ~~strip~~ **stripe** units $B_{y,1}$ - $B_{y,4}$, it being understood that the first error correction data need not be limited thereto. Parity of each stripe unit SP_y is typically calculated by logically combining data of stripe units $B_{y,1}$ - $B_{y,4}$. This logical combination is typically accomplished by exclusively ORing (XORing) data of the stripe units $B_{y,1}$ - $B_{y,4}$.

Please **replace** paragraph **[0025]** with the following amended paragraph:

[0025] Each stripe S_y has a corresponding entry of second error correction data stored in disk drive 16(6). The second error correction data may take any one of many different forms. In Figure 2, each entry of second error correction data is stored in components $P_{y,1}$ - $P_{y,4}$. Each component $P_{y,x}$ corresponds to a respective stripe unit $B_{x,y}$. In one embodiment, each component $P_{y,x}$ stores parity data calculated as a function of data in ~~strip~~ **stripe** unit $B_{x,y}$. Parity data of each component $P_{y,x}$ is typically calculated by logically combining data of stripe unit $B_{x,y}$. This logical combination is typically accomplished by exclusively ORing (XORing) data of stripe unit $B_{x,y}$. It is noted that each component $P_{y,x}$ need not store parity data. However, each component $P_{y,x}$ stores first error correction data generated by applying a particular algorithm to data of stripe unit $B_{x,y}$.

Please **replace** paragraph **[0030]** with the following amended paragraph:

[0030] RAID controller 18 receives a request for data stored in stripe unit $B_{y,x}$. In response to receiving the request in step 30, RAID controller reads existing data D_{old} of stripe unit $B_{y,x}$ and existing parity P_{old} of component $P_{y,x}$ as shown in step 31. Thereafter, in step 32, RAID controller 18 generates new parity P_{new} as a function of the ~~existingt~~ **existing** data D_{old} . The algorithm used to generate new parity P_{new} is the same algorithm used to generate existing parity data P_{old} in $P_{y,x}$ of disk 16(6). In step 34, RAID controller 18 compares the newly generated parity P_{new} with the existing parity P_{old}. If the newly generated parity P_{new} does not compare equally to existing parity P_{old}, then it is presumed that either the data (i.e, existing data D_{old}

within stripe unit $B_{y,x}$) sought by the read request is corrupted, the existing parity P_{old} of $P_{y,x}$ was generated invalidly, or other corruption problems exist. When newly generated parity P_{new} compares equally to existing parity P_{old}, the process proceeds to step 47 where data of stripe unit $B_{y,x}$ is returned to computer system 12. If in step 34 RAID controller 18 determines inequality between P_{new} and P_{old}, then the process proceeds to step 35 where RAID controller reads data of stripe units $B_{y,1}$-$B_{y,4}$, other than $B_{y,x}$, and parity data of stripe unit SP_y. Thereafter, RAID controller executes steps 36 and 37. In step ~~step~~ 36, RAID controller generates new data D_{new}. More particularly, RAID controller 18 generates new data D_{new} as a function of data of stripe units $B_{y,1}$-$B_{y,4}$, other than $B_{y,x}$, and parity data of stripe unit SP_y. In step 37, RAID controller generates second new parity data P'_{new} as a function of new data D_{new}. The algorithm used to generate second new parity data P'_{new} is the same algorithm used to generate new parity P_{new}. In step 38, RAID controller 18 compares second new parity data P'_{new} with new parity P_{new}. If second new parity data P'_{new} compares equally with new parity P_{new}, then existing data D_{old} in stripe unit $B_{x,y}$ is overwritten with the newly generated data D_{new}. However, if second new parity data P'_{new} does not compare equally with new parity P_{new} in step 38, the process proceeds to step 40 where RAID contoller 18 compares new data D_{new} with the old data D_{old}. If D_{new} does not compare equally with the old data D_{old}, then in one embodiment, an error message is returned to computer system 12 indicating that storage system 10 contains too much corruption.